Analysis of Cycle Stealing with Switching Times and Thresholds

Takayuki Osogami, * Mor Harchol-Balter [1]

Computer Science Department, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213

Alan Scheller-Wolf

Tepper School of Business, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213

Abstract

We consider two processors, each serving its own M/GI/1 queue, where one of the processors (the "donor") can help the other processor (the "beneficiary") with its jobs, during times when the donor processor is idle. That is the beneficiary processor "steals idle cycles" from the donor processor. There is a switching time required for the donor processor to start working on the beneficiary jobs, as well as a switching back time. We also allow for threshold constraints on both the beneficiary and donor sides, whereby the decision to help is based not only on idleness but also on satisfying threshold criteria in the number of jobs.

We analyze the mean response time for the donor and beneficiary processors. Our analysis is approximate, but can be made as accurate as desired, and is validated via simulation. Results of the analysis illuminate principles on the general benefits of cycle stealing and the design of cycle stealing policies.

Key words: Cycle stealing, load sharing, distributed system, matrix analytic, dimensionality reduction, Markov chain, threshold.
PACS:

* Corresponding author
Email addresses: osogami@cs.cmu.edu (Takayuki Osogami,), harchol@cs.cmu.edu (Mor Harchol-Balter), awolf@andrew.cmu.edu (Alan Scheller-Wolf).
URLs: http://www.cs.cmu.edu/~osogami/ (Takayuki Osogami,), http://www.cs.cmu.edu/~harchol/ (Mor Harchol-Balter).
[1] This work was supported by NSF Career Grant CCR-0133077, by NSF ITR Grant 99-167 ANI-0081396, and by IBM via PDG Grant 2003.

1 Introduction

Motivation Since the invention of networks of workstations, systems designers have touted the benefits of allowing a user to take advantage of machines other than her own, at times when those machines are idle. This notion is often referred to as *cycle stealing*. Cycle stealing allows such a user, Betty, with multiple jobs, to offload one of her jobs to the machine of a different user, Dan, if Dan's machine is idle, giving Betty two machines to process her jobs. When Dan's workload resumes, Betty must return to using only her own machine. We refer to Betty as the *beneficiary*, to her machine as the beneficiary machine/server, and to her jobs as beneficiary jobs. Likewise, we refer to Dan as the *donor*, to his machine as the donor machine/server, and to his jobs as donor jobs.

Although cycle stealing provides obvious benefits to the beneficiary, these benefits come at some cost to the donor. For example, the beneficiary's job may have to be checkpointed and the donor's working set memory reloaded before the donor can resume, delaying the resumption of processing of donor jobs. In our model we refer to these additional costs associated with cycle stealing as *switching times*.

A primary goal of this paper is to understand the benefit of cycle stealing for the beneficiary and the penalty to the donor, as a function of switching times. A secondary goal is to derive parameter settings for cycle stealing. In particular, given non-zero switching times, cycle stealing may pay only if the beneficiary's queue is "sufficiently" long. We seek to understand the optimal threshold on the beneficiary queue when switching to help, and the optimal threshold on the donor queue when switching back. More broadly, we seek general insights into which system parameters have the most significant impact on the effectiveness of cycle stealing.

The analytical model We assume there are two queues, the *beneficiary* queue and the *donor* queue, with independent arrival processes and service time distributions operating as M/GI/1/FCFS queues. Jobs arrive at rate λ_B (respectively, λ_D) at the beneficiary (respectively, donor) queue and have service requirement X_B with distribution G_B (respectively, X_D with distribution G_D). The load made up by beneficiary (respectively, donor) jobs is denoted by ρ_B (respectively, ρ_D) where $\rho_B = \lambda_B \cdot E[X_B]$ and $\rho_D = \lambda_D \cdot E[X_D]$. If the donor server is idle, *and* if the number of jobs at the beneficiary queue is at least N_B^{th}, the donor transitions into the switching state, for a random amount of time, K_{sw}. After K_{sw} time, the donor server is available to work on the beneficiary queue and the beneficiary queue becomes an M/G_B/2 queue. When the number of donor jobs in queue reaches N_D^{th} (either during K_{sw}, or during

the time the donor is helping the beneficiary), the donor transitions into a switching back state, for a random amount of time, K_{ba}. After the completion of the switch back, the donor server resumes working on its own jobs until the donor queue is empty. The donor server cannot work on any job while the donor is in the switching or switching back state.

A few details are in order. First, in the above model, the donor processor continues to cooperate with the beneficiary even if there is no beneficiary work left for it to do — the donor processor can switch back only when a donor job arrives. [2] Second, we assume that if the donor processor is working on a beneficiary job and a donor job arrives, that beneficiary job is returned to the beneficiary queue and will be resumed by the beneficiary processor as soon as that processor is available. The work done on the job is not lost (i.e. preemptive resume). [3] Our performance metric throughout is mean response time (overall and for each class of jobs), where the response time of a job is the time from when the job arrives until it completes service. We assume the first three moments of the service times are finite, and queues are stable.

Difficulty of analysis and previous work Consider the simplest instance of our problem — where the service requirements of all jobs are drawn from exponential distributions and the switching times and thresholds are zero. Even for this simplest instance the continuous time Markov chain, while easy to describe, appears computationally intractable. This is due to the fact that the stochastic process having state (*number of beneficiary jobs, number of donor jobs*) grows infinitely in two dimensions and contains no structure that can be easily exploited in practice to obtain an exact solution. While solution by truncation of the Markov chain is possible, the errors that are introduced by ignoring portions of the state space (infinite in two dimensions) can be significant, especially at higher traffic intensities. Thus truncation is neither sufficiently accurate nor robust for our purposes.

To our knowledge, there has been no previous analytical work on cycle stealing with switching times and thresholds. The analysis of cycle stealing *without* switching times and thresholds under exactly our model has been studied by

[2] We also analyzed the case where the donor processor switches back when it is not needed, see [11]. We found that this has almost no effect on performance, even under long switching times.

[3] It is easy to generalize our analysis to the case where the beneficiary requires a "switching time" before it can resume a job that the donor started. It is also trivial to extend our results to the case where all work on the job in progress is lost if a donor job arrives, provided that we assume that the job is restarted with a new service time — which we feel is unrealistic. It is also possible to extend our results to the case where the donor must complete the beneficiary job in progress before it switches back (see Section 7).

Foley and McDonald [5]; they prove asymptotic, heavy-traffic bounds on the performance of cycle stealing under exponential job size distributions.

A related model to our cycle stealing model is the "coupled processor model," which we elaborate below. In this model two processors each serve their own class of jobs, and if either is idle it may help the other, increasing the rate of the other processor. This help incurs no switching time and has a benefit even if only a single job is present (i.e. two processors can work on the single job). However, because the processors work in concert, rather than on different jobs, these systems will gain no multi-server benefit when serving highly variable jobs; short jobs may get stuck waiting behind long jobs in the single queue for each class. All works we mention below consider Poisson arrivals.

Early work on the coupled processor model includes papers by Fayolle and Iasnogorodski [4] and Konheim, Meilijson and Melkman [8]. Both papers assume exponential service times, deriving expressions for the stationary distribution of the number of jobs of each type. Fayolle and Iasnogorodski use complex algebra, eventually solving either a Dirichlet boundary value problem or a Riemann-Hilbert boundary value problem, depending on the accelerated rates of the servers. Konheim et. al. assume that the accelerated rate is twice the original rate, which yields simpler expressions (still in the form of complex integrals). While it is possible to numerically evaluate these analytical expressions, they were not evaluated in either work; thus no intuition was provided on the performance of these systems.

The above work is extended by Cohen and Boxma [3] to the case of general service times. They consider stationary *workload*, which they formulate as a Wiener-Hopf boundary value problem. This leads to expressions involving either integrals or infinite sums; if the queues are symmetric simpler expressions for mean total workload are found, but not for mean *response* time. They again have the two processors working in concert, without a switching time, providing analytical expressions, rather than numerical values.

In more recent work, Borst, Boxma and van Uitert [2] apply a transform method to the expressions in [3], yielding asymptotic relations between the workloads and the service requirement distributions. This leads to the insight that if a processor has a load less than one, it is "insulated" from the heavy-tail of the other, as long periods without cooperation will not lead to large backlogs. This is not the case if the load is greater than one, as the queue now must rely on help to be stable. Borst, Boxma and Jelenkovic [1] consider a similar question under a related model of generalized processor sharing, where n classes of jobs can share a processor with arbitrary weights. Using probabilistic bounds, they show that different service rates can either insulate the performance of different classes from the others or not, again depending on whether the non-cooperative load is larger than one. Both of these papers

are concerned with the asymptotic behavior of workload, whereas our work isolates mean response time. Our work is thus complementary to these results.

Our approach This paper presents the first analysis of cycle stealing under general service requirements with switching times and thresholds. Recall that the difficulty in analyzing cycle stealing is that the corresponding stochastic process has state space that grows infinitely in two dimensions (2D), making it computationally intractable. The key idea in our approach is to find a way to transform this 2D-infinite Markov chain into a Markov chain that is infinite in only one dimension (1D-infinite Markov chain), which can be analyzed efficiently. The questions in applying such a transformation are (i) what should the 1D-infinite Markov chain track, and (ii) how can all the relevant information from the 2D chain be captured in the 1D-infinite Markov chain. Our 1D-infinite Markov chain tracks the number of beneficiary jobs, yielding measures of beneficiary performance. For the donor jobs, our state space contains only limited knowledge, 0, 1, ..., N_D^{th}, or $\geq N_D^{th}$ jobs. Nevertheless we are able to capture all necessary information by using special transitions in our Markov chain, where these transitions represent the lengths of an assortment of busy periods. We refer to this technique as *dimensionality reduction*. The difficulty in dimensionality reduction lies in specifying the right busy periods, some of which transcend the definition of the analytical model.

Once the 1D-infinite Markov chain is specified, the hard work is finished, since the limiting probabilities in this chain can be solved efficiently using known numerical (matrix analytic) techniques, which in turn gives mean response time for beneficiary jobs. The mean response time of donor jobs is analyzed as an M/GI/1 queue with generalized vacations, and all the necessary information is provided by the limiting probabilities of the 1D-infinite Markov chain. While a closed-form solution may be preferable, our chain is compact enough, and matrix analytic methods powerful enough, that only a couple of seconds are required to generate most of the results plots in this paper. Furthermore, our method generalizes to more complex models, e.g. multiple donors (Section 7).

Our analysis is approximate but can be made as accurate as desired. The primary approximation lies in representing the length of the busy periods by phase type (PH) distributions. The beneficiary service requirement (X_B) and the switching time to help (K_{sw}) must also be approximated by PH distributions, although X_D and K_{ba} can be general. In this paper, we use a PH distribution, shown in Figure 1, to capture the first 3 moments of the busy periods, and verify that this is sufficient via simulation (See Section 5) [4]. For X_B and K_{sw}, we assume throughout the paper that they have PH distributions, and hence no approximation is involved.

[4] Our method naturally extends to matching more moments.

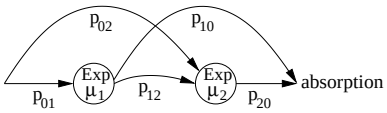


Fig. 1. *A 2-phase PH distribution with Coxian representation. Notice that a PH distribution is the distribution of the absorption time in a continuous time Markov chain. The ith state has exponentially-distributed sojourn time with rate μ_i. With probability p_{0i} we start in the ith state, and the next state is state j with probability p_{ij}. Each state has some probability of leading to absorption. The absorption time is the sum of the times spent in each of the states.*

Summary of results Our analysis yields many interesting results concerning cycle stealing, detailed in Sections 4 and 6. While cycle stealing obviously benefits the beneficiaries (beneficiary jobs) and hurts the donors (donor jobs), we find that when $\rho_B > 1$, cycle stealing is profitable overall even under significant switching times, as it may ensure stability of the beneficiary queue. For $\rho_B < 1$, we define load-regions under which cycle stealing pays. We find that in general the switching time is only prohibitive when it is large compared with $E[X_D]$. Under zero switching time, cycle stealing always pays.

Two counterintuitive results are that when $\rho_B < 1$, the mean response time of the beneficiaries is surprisingly insensitive to the switching time, and also insensitive to the variability of the donor job size distribution. Even when the variability of the donor job sizes is very high, and donor help thus is very bursty, the beneficiaries still enjoy significant benefits.

Our analysis also allows us to investigate characteristics of the beneficiary and donor side thresholds, N_B^{th} and N_D^{th}, both with respect to their impact on stability and their impact on mean response time. With respect to beneficiary stability, we find that N_B^{th} has no effect, while increasing N_D^{th} increases the stability region. Donor stability is not affected by either threshold. With respect to overall mean response time, we find that mean response time is far more sensitive to changes in N_D^{th} than to changes in N_B^{th}. This extends the results of [12], where we study only the effect that N_B^{th} has on mean response time. We find the optimal value of N_B^{th} tends to be well above 1. The reason is that increasing N_B^{th} does not appreciably diminish beneficiary gain, but it does alleviate donor pain. We find that the optimal setting of N_B^{th} is an increasing function of ρ_B, ρ_D, and switching times. By contrast, we find that the optimal value of N_D^{th} is often close to 1, provided $\rho_B < 1$. Increasing N_D^{th} significantly hurts the donor, although it may provide significant help to the beneficiary if ρ_B is high. We find that the optimal N_D^{th} is *not* a monotonic function of ρ_D, but is an increasing function of ρ_B and switching times.

Our model deals with switching times in a general way, making the results both applicable to a shared-memory multiprocessor architecture and to a network of workstations (NOW). Our switching times, K_{sw} and K_{ba}, can be viewed as

the time for switching between job types in a shared-memory multiprocessor architecture. In a NOW, there is an additional overhead incurred from migrating jobs from one server to another, where jobs which have not started running require negligible overhead, whereas migrating a "running" job (in progress) requires high overhead, since all of its state must be migrated with it. Our model captures this notion in that the switching back time, K_{ba}, can be viewed as the time incurred for preempting an already running job and returning it to the beneficiary.

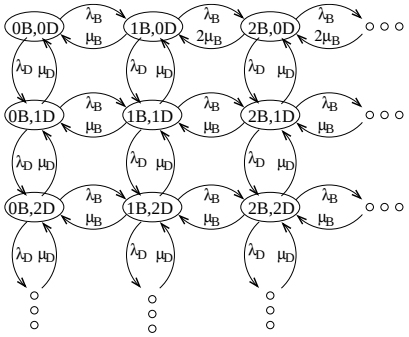
2 Analysis of beneficiary mean response time

In this section we analyze the mean response time of beneficiary jobs. The key idea in the analysis is to reduce a 2D-infinite Markov chain to a 1D-infinite Markov chain. In Section 2.1 we illustrate this dimensionality reduction technique via the analysis of the simplest case with zero switching times (i.e. $K_{sw} = K_{ba} = 0$) and threshold values fixed at 1 (i.e. $N_B^{th} = N_D^{th} = 1$). In Section 2.2 we extend the analysis to the general case of nonzero switching times and arbitrary threshold values.

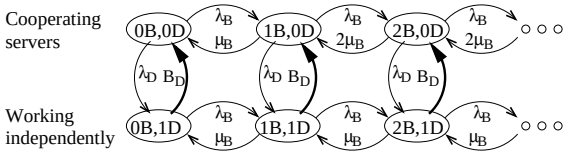
2.1 Dimensionality reduction

In this section we introduce the dimensionality reduction technique via the analysis of the simplest case of zero switching times and threshold values fixed at 1. We first assume that the job sizes are exponentially distributed, and then show how the analysis can be extended to the case of PH job sizes.

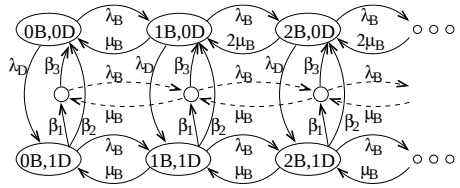
Figure 2(a) shows the 2D-infinite Markov chain which tracks the number of beneficiary and donor jobs in the case where all job sizes have exponential distributions. Figure 2(b) shows a transition diagram with 1D-infinite state space, which corresponds to Figure 2(a). Observe that this 1D-infinite transition diagram exactly tracks the number of beneficiary jobs, while only providing binary information on the number of donor jobs (specifically whether there are zero or at least one donor job). This transition diagram is *not* a Markov chain, since the sojourn time in the bottom row is the time from when the donor server has at least one job until the donor server is free, namely a donor busy period, B_D. Figure 2(c) shows the 1D-infinite Markov chain where B_D is represented by a 2-phase PH distribution with Coxian representation (see Figure 1). The parameters for the PH distribution are chosen to match the first three moments of B_D (see [10]). The moments of B_D are obtained by differentiating its Laplace transform, $\widetilde{B_D}(s) = \widetilde{X_D}(s + \lambda_D - \lambda_D \widetilde{B_D}(s))$, where $\widetilde{X_D}(\cdot)$ denotes the Laplace transform of the distribution of X_D.



(a) 2D-infinite Markov chain



(b) 1D-infinite transition diagram



(c) 1D-infinite Markov chain

Fig. 2. *Markov chains for cycle stealing, where* $N_B^{th} = N_D^{th} = 1$, $K_{sw} = K_{ba} = 0$, *and* X_B *is exponentially distributed with rate* μ_B. *Figure (a) shows a 2D-infinite Markov chain tracking both the number of beneficiary jobs and the number of donor jobs. Figure (b) shows a 1D-infinite transition diagram tracking the number of beneficiary jobs and binary information (zero or* ≥ 1*) on the number of donor jobs, where* B_D *is represented by a bold transition. Figure (c) shows a 1D-infinite Markov chain, where* B_D *is represented by a 2-phase PH distribution with Coxian representation.*

So far we assumed that the job sizes have exponential distributions, but this can be generalized. The donor job size, X_D, is used only to calculate $\widetilde{B_D}(s)$, and this is valid for general distributions. The beneficiary job size, X_B, can also be generalized to a PH distribution. Figure 3 shows how the beneficiary job size is replaced by a 2-phase PH distribution with Coxian representation. Observe that the state space now must maintain states for one or two partially-completed beneficiary jobs. If a donor job arrives while the Markov chain is in the cooperating row, the job that the donor was working on will be moved to the head of the beneficiary queue (we currently assume zero time for the transfer, but this is easy to generalize). To make this transition diagram to a 1D-infinite Markov chain, the bold transitions in Figure 3 should be replaced by PH distributions as we do in Figure 2(c).

The mean response time for beneficiary jobs can be computed via Little's law once we know the mean number of jobs. The number of beneficiary jobs is tracked in the 1D-infinite Markov chains, and the limiting probabilities can be analyzed using the matrix analytic method. The matrix analytic method is particularly efficient when it is applied to a QBD (quasi-birth-and-death) process with 1D-infinite state space as in Figure 2(c). We use algorithms presented in Sections 8.4 and 12.2 of [9]. Most of the plots in this paper were produced within a couple of seconds using Matlab 6 running on Linux, on a 1 GHz Pentium III with 512 MB RAM.

We emphasize the simplicity of our approach. It is easy to establish a 1D-infinite Markov chain. It is also easy to extend the job size distributions to PH distributions. In Section 2.2, we analyze the more general case: nonzero

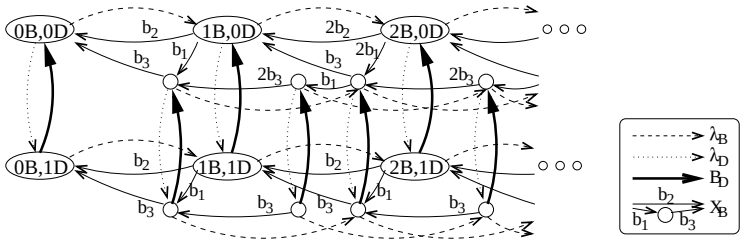


Fig. 3. *A transition diagram for cycle stealing where* $N_B^{th} = N_D^{th} = 1$, $K_{sw} = K_{ba} = 0$, *and* X_B *has a 2-phase PH distribution with Coxian representation (Figure 1).* B_D *is represented by a bold transition, but this should be replaced by a PH distribution as in Figure 2(c).*

switching times and arbitrary threshold values. We will see that various types of busy periods are needed to apply the the dimensionality reduction technique in the general case, but we are still able to model the system using a 1D-infinite Markov chain that can be analyzed efficiently via the matrix analytic method.

Notice also that approximating general distributions (of busy period durations and of job sizes) by PH distributions is the only approximation in our approach, and this can be made as accurate as desired by using more phases and matching more moments. However, we find that matching the first three moments of the distribution provides sufficient accuracy in the analysis of the mean response time (see Section 5).

2.2 Analysis of the general case

In this section we illustrate the analysis for the general case: nonzero switching times and arbitrary threshold values. We assume that the job sizes and switching times have exponential distributions, but this can be generalized using PH distributions in the same way as in Section 2.1. In the analysis of the simplest case in Section 2.1, the only information that we need to track about the donor is whether the number of the donor jobs is zero or ≥ 1. In the analysis of the general case, we need additional information: whether the donor is in the process of switching, and more information on the number of donor jobs (0, 1,..., $N_D^{th} - 1$, or $\geq N_D^{th}$). We first describe these additional states. We then apply the dimensionality reduction technique introduced in Section 2.1 and obtain a 1D-infinite Markov chain for the particular case of $N_B^{th} = 3$ and $N_D^{th} = 2$. Alternative threshold values can be handled similarly. This 1D-infinite Markov chain can be analyzed via the matrix analytic method in the same way as in Section 2.1.

We define $2N_D^{th} + 3$ states (regions) of the donor that we track in the 1D-infinite Markov chain. These regions are denoted by I_0, I_{1+}, B, C_i, and S_i for $i = 0, 1, ..., N_D^{th} - 1$ (see Figure 4(a)). In region I_0 the donor server is residing

at the donor queue and the number of donor jobs is zero. In region I_{1+} the donor server is working on donor jobs independently of the beneficiary server and there is at least one donor job. When we are in region I_0, a donor arrival triggers a transition to region I_{1+}; however, if the number of beneficiary jobs becomes N_B^{th} due to a beneficiary arrival, we transition to region S_0, where the donor server is in the process of switching to the beneficiary queue and the number of donor jobs is 0. The sojourn time in region I_{1+} is the length of the donor busy period started by a donor job; so the donor queue is empty when we leave this region. After this busy period, we transition back to region I_0 if the number of beneficiary jobs is $< N_B^{th}$; otherwise, we transition to region S_0.

In regions S_i $(i = 0, ..., N_D^{th} - 1)$ the donor server is in the process of switching to the beneficiary queue and the number of donor jobs is i. In region S_i we either have a donor job arrival, which triggers a transition to region S_{i+1} (or region B, which we define below, if $i = N_D^{th} - 1$) for $i = 0, ..., N_D^{th} - 1$, or complete the switching, which triggers a transition to region C_i. In regions C_i $(i = 0, ..., N_D^{th} - 1)$ the donor server is working on the beneficiary jobs (cooperating with the beneficiary server) and the number of donor jobs is i. When we are in region C_i, a donor job arrival causes a transition to region C_{i+1} (or region B if $i = N_D^{th} - 1$) for $i = 0, ..., N_D^{th} - 1$.

In region B the donor server is in the process of switching back since there are N_D^{th} donor jobs. We define the time spent in region B to include both K_{ba} *and* the time to empty the donor queue; i.e. the sojourn time in region B is the length of the donor busy period started by N_D^{th} donor jobs and K_{ba}. After this busy period, we transition back to region I_0 if the number of beneficiary jobs is $< N_B^{th}$, and transition to region S_0 otherwise.

Figure 4(b) shows the transition diagram with 1D-infinite state space for the case where $N_B^{th} = 3$ and $N_D^{th} = 2$. This state space tracks the exact number of beneficiary jobs. Each row corresponds to one of the regions, I_0, I_{1+}, B, C_i, and S_i for $i = 0$ or 1, tracking the information needed about the donor. In applying the dimensionality reduction technique, we introduce two types of busy period transitions, B_D and B_{2D+ba}. B_D represents the donor busy period started by a donor job, and B_{nD+ba} represents the donor busy period started by n donor jobs ($n = 2$ in this case) and switching back time, K_{ba}. These busy periods are represented by bold transitions. To create the 1D-infinite Markov chain from this 1D-infinite transition diagram, we replace the bold transition by a PH distribution that matches the first three moments of the busy period duration. The moments of B_D are computed by differentiating $\widetilde{B_D}(s)$, and the moments of B_{nD+ba} are computed by differentiating the Laplace transform: $\widetilde{B_{nD+ba}}(s) = \widetilde{K_{ba}}(s + \lambda_D - \lambda_D \widetilde{B_D}(s)) \cdot \left(\widetilde{X_D}(s + \lambda_D - \lambda_D \widetilde{B_D}(s))\right)^n$.

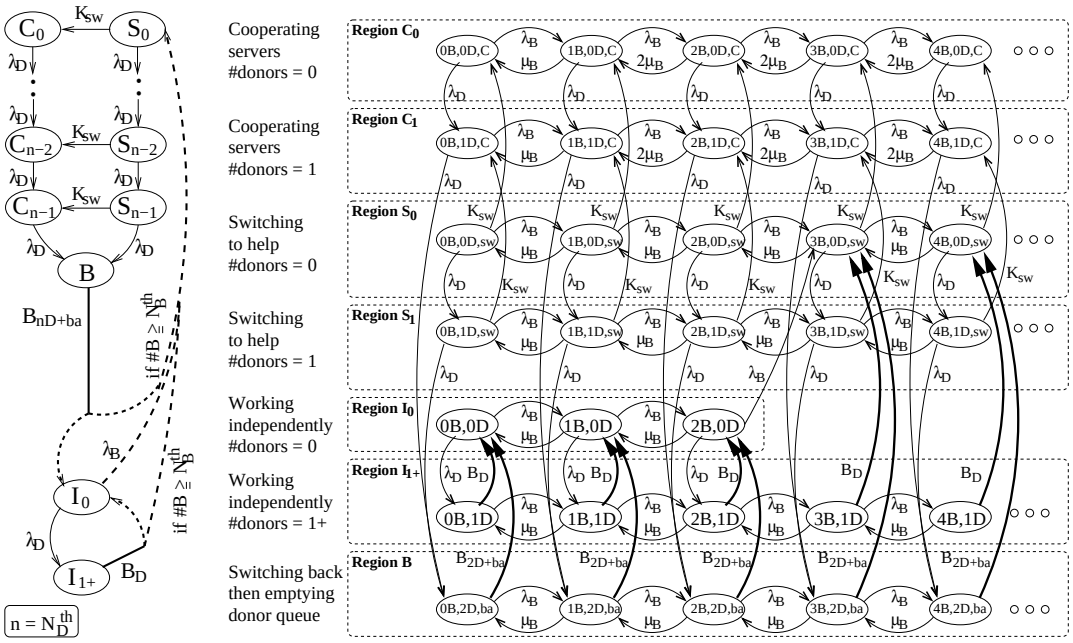


Fig. 4. *Transition diagrams used in the analysis of beneficiary mean response time in the general case. Column (a) shows a transition diagram among regions, I_0, I_{1+}, B, C_i, and S_i for $i = 0, ..., N_D^{th} - 1$ for the general case of cycle stealing. Column (b) shows a transition diagram for cycle stealing where $N_D^{th} = 2$, $N_B^{th} = 3$, and X_B and K_{sw} have exponential distributions. Busy periods, B_D and B_{2D+ba}, are drawn using a single transition, but this should be replaced by PH distributions as in Figure 2(c).*

3 Analysis of donor mean response time

In this section we analyze the mean response time of donor jobs. The donor queue is analyzed as an M/GI/1 queue with generalized vacations [6], where we use the limiting probabilities that we compute in Section 2. The following theorem gives a way to calculate the mean response time of donor jobs:

Theorem 1 *The mean response time of donor jobs, $E[T_D]$, is given by*

$$E[T_D] = E[X_D] + \frac{\lambda_D E[X_D^2]}{2(1-\rho_D)} + \frac{\left(N_D^{th}(N_D^{th}-1) + 2N_D^{th}\lambda_D E[K_{ba}] + \lambda_D^2 E[K_{ba}^2]\right) p}{2\lambda_D\left((N_D^{th} + \lambda_D E[K_{ba}])p + (1-p)\right)},$$

where

$$p = \frac{Pr(Region\ C_{N_D^{th}-1}) + Pr(Region\ S_{N_D^{th}-1})}{Pr(Region\ C_{N_D^{th}-1}) + Pr(Region\ S_{N_D^{th}-1}) + Pr(Region\ I_0)}, \tag{1}$$

where $\Pr(\text{Region } R)$ *is the limiting probability that the system state is in region R for $R = I_0, C_{N_D^{th}-1}$, and $S_{N_D^{th}-1}$.*

Proof: The donor queue can be seen as an M/GI/1 queue with generalized vacations, where a vacation starts when the number of donor jobs becomes zero and ends when the donor server starts working on donor jobs. In an M/GI/1 queue with generalized vacations, the mean response time has the following expression [6]:

$$E[T] = E[X_D] + \frac{\lambda_D E[X_D^2]}{2(1-\rho_D)} + \frac{E[A(A-1)]}{2\lambda_D E[A]}, \tag{2}$$

where A denotes the number of jobs that arrive during a vacation period. Observe that the first two terms constitute the mean response time in a corresponding M/GI/1 queue (without vacation). Therefore, it suffices to analyze $E[A]$ and $E[A(A-1)]$.

There are two types of vacations, depending on how the vacation ends. The first type of vacation ends when a donor job arrives at an empty donor queue while the donor server is staying at the donor queue. In this case, $A = 1$. The second type of vacation ends when a donor job arrives at a donor queue with $N_D^{th}-1$ jobs while the donor server is staying at the beneficiary queue or in the process of switching to the beneficiary queue. In this case, $A = N_D^{th} + B$, where B is the number of donor job arrivals during K_{ba}. Let p be the probability that a vacation is of the second type. Then,

$$E[A] = (N_D^{th} + \lambda_D E[K_{ba}])p + (1-p) \tag{3}$$

$$E[A(A-1)] = \left(N_D^{th}(N_D^{th}-1) + 2N_D^{th}\lambda_D E[K_{ba}] + \lambda_D^2 E[K_{ba}^2]\right) p. \tag{4}$$

All that remains is to derive p. Observe that the first type of vacation starts when a donor job arrives while the system state is in Region I_0, and the second type of vacation starts when a donor job arrives while the system state is either in Region $C_{N_D^{th}-1}$ or in Region $S_{N_D^{th}-1}$. Since the arrival process is Poisson, p is given by the expression (1). The theorem now follows from (2)-(4). □

4 Stability

In this section we derive stability conditions for cycle stealing with switching times and thresholds. We find for example that the stability condition for the donor jobs remains $\rho_D < 1$, regardless of whether the donor jobs experience switching times. By contrast, the stability region for the beneficiary jobs can be significantly below $2 - \rho_D$, specifically because the switching time erodes the beneficiary stability region. Also, interestingly, the value of N_B^{th} does not affect the stability region of either the donor or beneficiary jobs. By contrast, increasing N_D^{th} increases the stability region of the beneficiary jobs; however it has no effect of the stability region of the donor jobs.

Theorem 2 *The donor queue is stable iff* $\rho_D < 1$.

Proof: It makes intuitive sense that the donor queue is stable iff $\rho_D < 1$ regardless of the switching times and threshold values, since the donor would never switch if the donor queue was persistently backlogged. Below, we prove sufficiency more formally. The necessity of $\rho_D < 1$ is trivial.

Assume $\rho_D < 1$. Let B_D denote the length of a busy period at the donor queue. We first consider the case of $N_B^{th} = 0$. A busy period at the donor queue is started by a switching time K_{ba} and N_D^{th} donor jobs. As $\rho_D < 1$, the mean length of a busy period is $E[B_D] = \frac{N_D^{th} E[X_D] + E[K_{ba}]}{1-\rho_D} < \infty$. In this case B_D clearly has a proper limiting distribution, and hence the donor is stable. Next we consider the case of $N_B^{th} > 0$. In this case $E[B_D]$ is smaller than in the case of $N_B^{th} = 0$ because there will be donor busy periods in which the donor hasn't left the donor queue, implying (i) there is no switching back time, and (ii) the busy period is started by only one donor job. □

Before we derive the stability condition on ρ_B, we prove a lemma allowing us to assume $N_B^{th} = 0$.

Lemma 1 *If the beneficiary queue is stable at* $N_B^{th} = 0$*, then it is stable at* $N_B^{th} = n$, $\forall\ 0 \leq n < \infty$.

Proof: Intuitively, the stability of the beneficiary queue is independent of N_B^{th}, since N_B^{th} would be irrelevant when the beneficiary queue was continuously backlogged.

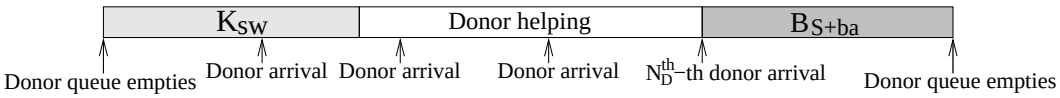
More formally, let $L_B^{(n)}(t)$ denote the number of beneficiary jobs at time t given $N_B^{th} = n \geq 1$. Consider a new process $\widehat{L}_B^{(n)}(t)$ in which the number of jobs at time $t = 0$ is n, instead of 0 as in the original process, and no service is given by either server to a beneficiary job if there are $\leq n$ jobs present at the beneficiary queue. Note that $\widehat{L}_B^{(n)}(t)$ stochastically dominates $L_B^{(n)}(t)$. Also, along any sample path, $\widehat{L}_B^{(n)}(t)$ will be equal to $n + L_B^{(0)}(t)$. Therefore, if $L_B^{(0)}(t)$ is proper, so too is $\widehat{L}_B^{(n)}(t)$ and hence $L_B^{(n)}(t)$. □

We now prove the stability condition on ρ_B.

Theorem 3 *The beneficiary queue is stable iff*

$$\rho_B < 1 + \frac{\max(1-\rho_D, 0) \sum_{i=0}^{N_D^{th}} (N_D^{th} - i) \frac{(-\lambda_D)^i}{i!} \tilde{K}_{sw}^{(i)}(\lambda_D)}{N_D^{th} + \lambda_D E[K_{ba}]}, \tag{5}$$

where $\tilde{K}_{sw}(s)$ *is the Laplace transform of* K_{sw} *and* $\tilde{K}_{sw}^{(i)}(s)$ *is its* i*-th derivative. In particular, when* K_{sw} *is exponentially distributed, the condition is expressed*



by the closed form:

$$\rho_B < 1 + \frac{\max(1-\rho_D, 0)\left\{N_D^{th} - \frac{q}{1-q}\left(1-q^{N_D^{th}}\right)\right\}}{N_D^{th} + \lambda_D E[K_{ba}]}, \tag{6}$$

where $q = \frac{\lambda_D E[K_{sw}]}{1+\lambda_D E[K_{sw}]}$.

Proof: We first prove sufficiency. By Lemma 1, we can assume $N_B^{th} = 0$. Let F denote the fraction of time that the donor helps the beneficiary. Then the strong law of large numbers can be used to show that the necessary and sufficient condition for stability of the beneficiary jobs is $\rho_B < 1+F$. If $\rho_D \geq 1$, $F = 0$. Thus we assume $\rho_D < 1$ and derive F using renewal reward theory.

Let a renewal occur every time the donor queue becomes empty. Recall $N_B^{th} = 0$. By renewal-reward theory, the fraction of time donor helps beneficiary is $F = \frac{E[R]}{E[Y]}$, where R denotes the total time that donor helps beneficiary (reward) during the renewal cycle, and Y denotes the length of the renewal cycle. Observe that there may be any number of donor arrivals ranging from 0 to N_D^{th} during K_{sw} and we switch back only after the N_D^{th} arrival.

Let S denote the sum of the service times of N_D^{th} donor jobs and B_{S+ba} denote the length of the busy period started by these jobs of total size S plus K_{ba}. Then, as $\rho_D < 1$,

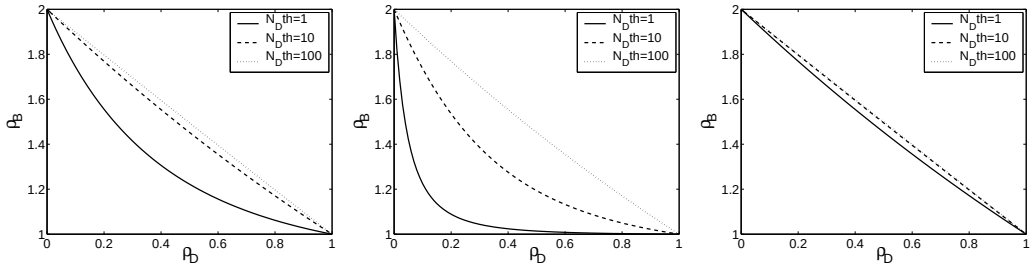
$$E[Y] = N_D^{th} \cdot E[I_D] + E[B_{S+ba}] = \frac{N_D^{th}}{\lambda_D} + \frac{N_D^{th} E[X_D] + E[K_{ba}]}{1-\rho_D},$$

where I_D is the interarrival time for donor jobs.

To compute $E[R]$ we condition on the number of donor arrivals during K_{sw}. If there are i arrivals, then the mean time spent helping is the time until there are $(N_D^{th} - i)$ more donor arrivals, $(N_D^{th} - i)E[I_D]$. Let p_i denote the probability that there are i donor arrivals during K_{sw}. Then, $p_i = \frac{(-\lambda_D)^i}{i!}\tilde{K}_{sw}^{(i)}(\lambda_D)$. Using p_i, $E[R]$ is now derived as follows:

$$E[R] = \sum_{i=0}^{N_D^{th}-1} (N_D^{th} - i)\frac{1}{\lambda_D} p_i.$$

Thus, $\rho_B < 1+F$ is equivalent to (5). When K_{sw} is exponentially-distributed, $p_i = q^i(1-q)$, where $q = \frac{\lambda_D}{\lambda_D + \mu_{sw}}$. Therefore,



(1) $E[X_D] = 1, E[K] = 1$ (2) $E[X_D] = 1, E[K] = 10$ (3) $E[X_D] = 10, E[K] = 1$

Fig. 5. *Stability condition on beneficiaries for cycle stealing with switching times ($K = K_{sw} = K_{ba}$) and thresholds.*

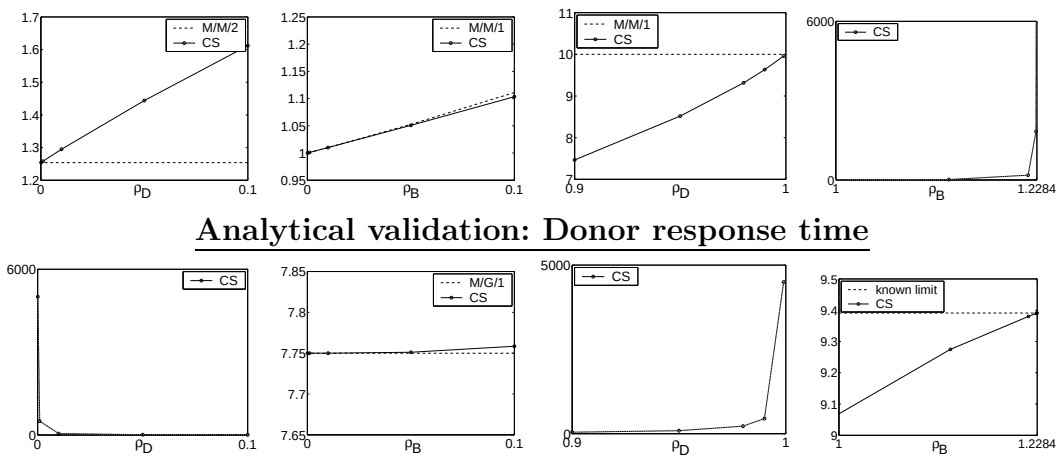
$$E[R] = \frac{1}{\lambda_D}\left\{N_D^{th} - \frac{q}{1-q}(1-q^{N_D^{th}})\right\}.$$

Thus, $\rho_B < 1 + F$ is equivalent to (6).

Above, we have proved the necessary and sufficient condition for $N_B^{th} = 0$. By Lemma 1, this is also the sufficient condition for $N_B^{th} > 0$. Now, we prove necessity for $N_B^{th} > 0$. When $N_B^{th} > 0$, the donor server does not necessarily help the beneficiary even when it is available for help. Therefore, there are two types of renewal periods. In the first type of renewal period, the donor server helps the beneficiary, i.e. $R > 0$. In this case, $E[Y]$ is the same as for $N_B^{th} = 0$, and $E[R]$ for $N_B^{th} > 0$ is at most $E[R]$ for $N_B^{th} = 0$. In the second type of renewal period, the donor server does not help the beneficiary, i.e. $R = 0$. (In this case $E[Y]$ can be smaller than that for $N_B^{th} = 0$.) The fraction of time, F, that the donor server helps the beneficiary can be expressed as $F = F_1 + F_2$, where F_1 is the fraction of time that the donor server helps the beneficiary and the renewal period is type 1, and F_2 is the fraction of time that the donor server helps the beneficiary and the renewal period is type 2. In fact, $F_2 = 0$. Therefore, $F = F_1 \leq \frac{E[R]}{E[Y]}$. This proves the necessity for $N_B^{th} > 0$. $\square$

Note that the right hand side of (5) is an increasing function of N_D^{th}; in terms of stability, the larger N_D^{th}, the more stable the beneficiary queue. In particular, when $N_D^{th} = 0$, (5) is $\rho_B < 1$; as $N_D^{th} \to \infty$, (5) becomes $\rho_B < 2 - \rho_D$.

Figure 5 shows the stability condition for beneficiaries as a function of ρ_D when K_{sw} is exponentially distributed. In case (1), we set $E[X_D] = 1$ and $E[K_{sw}] = E[K_{ba}] = 1$. The region below each line satisfies the stability condition. As N_D^{th} increases as high as 100, the effect of switching overhead becomes negligible, and the stability condition approaches $\rho_B < 2 - \rho_D$. In case (2), we set $E[X_D] = 1$ and $E[K_{sw}] = E[K_{ba}] = 10$. The switching time is large, and there is little benefit at moderate or high ρ_D in terms of stability, unless N_D^{th} is large. However, there is still large benefit at low ρ_D. In case (3), we set $E[X_D] = 10$ and $E[K_{sw}] = E[K_{ba}] = 1$. The stability region is much larger; when $N_D^{th} = 1$, the stability region is almost the same as that of $N_D^{th} = 10$



(a) $\rho_B = 0.9, \rho_D \to 0$ (b) $\rho_B \to 0, \rho_D = 0.6$ (c) $\rho_B = 0.9, \rho_D \to 1$ (d) $\rho_B \to \rho_B^{\max}, \rho_D = 0.6$

Fig. 6. *Validation of our analysis against four limiting cases, where* $N_B^{th} = 3$, $N_D^{th} = 2$, $E[K_{sw}] = E[K_{ba}] = 1$, X_B *is exponentially-distributed with mean 1, and* X_D *has a PH distribution with mean 1 and* $C_D^2 = 8$.

in case (1). This is intuitive: when $E[X_D] = 10$ and $N_D^{th} = 1$, the expected amount of donor work when the donor switches back is the same as that when $E[X_D] = 1$ and $N_D^{th} = 10$.

5 Validation of analysis

Since our analysis involves approximation of busy periods by PH distributions, it is of paramount importance to validate the analysis. Analytical validation against limiting load cases is presented in Section 5.1, and simulation validation over a range of load is reported in Section 5.2.

5.1 Validation against known limiting cases

We evaluate the performance of cycle stealing under four limiting situations: $\rho_D \to 0$, $\rho_B \to 0$, $\rho_D \to 1$, and $\rho_B \to \rho_B^{\max}$, where $\rho_B^{\max}$ is the right hand side of (5). We assume that $N_B^{th} = 3$, $N_D^{th} = 2$, $E[K_{sw}] = E[K_{ba}] = 1$, X_B is exponentially-distributed with mean 1, and X_D has a PH distribution with mean 1 and $C_D^2 = 8$. For these limiting cases, the mean response times of the beneficiaries and donors are easy to evaluate.

Figure 6 verifies that our analysis has the correct limiting behavior in all of these cases. In case (a), $\rho_B = 0.9$ and $\rho_D \to 0$. The mean response time for beneficiary jobs converges to that under an M/M/2 queue, since the donor

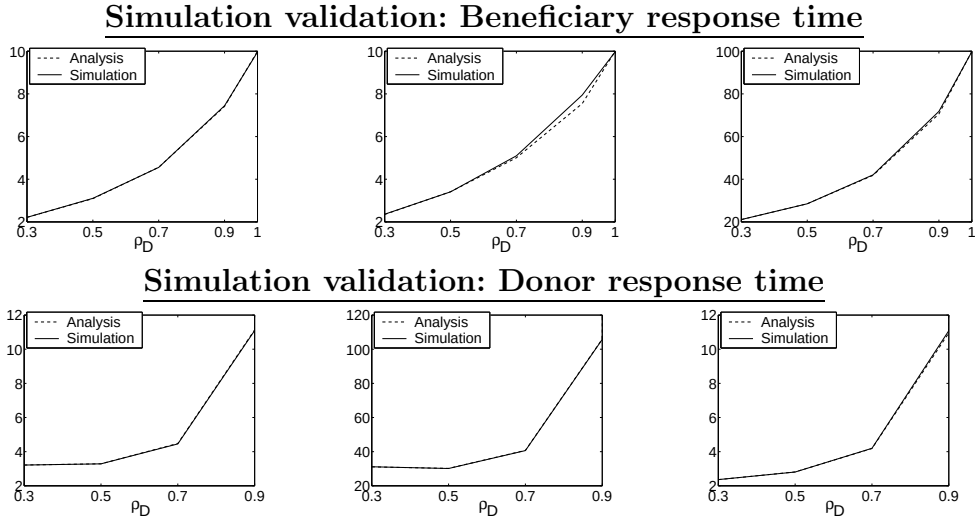
server can almost always help when $\rho_D \sim 0$. The mean response time for donor jobs diverges to infinity. This is because the donor job arriving at an empty donor queue must wait for the next arrival before being served (note $N_D^{th} = 2$), and the interarrival time diverges to infinity as $\rho_D \to 0$. In case (b), $\rho_D = 0.6$ and $\rho_B \to 0$. The mean response time for beneficiary jobs converges to that under an M/M/1 queue, since the number of beneficiary jobs is almost always $< N_B^{th}$ when $\rho_B \sim 0$ and hence there is no help from the donor server. The mean response time for donor jobs converges to that under an M/GI/1 queue, since the donor server is almost always at the donor queue and hence requires no setup time. In case (c), $\rho_B = 0.9$ and $\rho_D \to 1$. The mean response time for beneficiary jobs converges to that under an M/M/1 queue, since the donor server is almost always busy working on donor jobs when $\rho_D \sim 1$ and hence there is no help from the donor server for the beneficiary. The mean response time for donor jobs diverges to infinity, since $\rho_D < 1$ is the stability condition. In case (d), $\rho_D = 0.6$ and $\rho_B \to \rho_B^{\max}$. The mean response time for beneficiary jobs diverges to infinity as $\rho_B \to \rho_B^{\max}$, as is predicted by the stability condition. The mean response time of donor jobs converges to the expression in Theorem 1 with $p = 1$, since Pr(Region I_0)$\to 0$ as $\rho_B \to \rho_B^{\max}$.

5.2 Validation against simulation

The accuracy of our analysis is also validated against simulation: a subset of our validation experiments is shown in Figure 7. In simulation, 1,000,000 events are generated in each run. We assume that job sizes are exponential, $N_B^{th} = 3$, and $N_D^{th} = 2$. We show three cases: 7(a), $E[X_B] = E[X_D] = 1$; 7(b), $E[X_B] = 1$ and $E[X_D] = 10$; 7(c), $E[X_B] = 10$ and $E[X_D] = 1$. In all cases, the results of analysis are in very close agreement with simulation. The only mild discrepancy is the mean response time of the beneficiaries in case (b), under high load ($\rho_D = .9$). Under case (b), donor jobs are large, making their busy periods more variable, especially at high loads. As our analysis is very dependent on these busy periods, matching only the first three moments may introduce error in this case. We hypothesize that the accuracy of our analysis would improve if we matched more moments of the busy periods using PH distributions with more phases.

6 Results of Analysis

This section discusses our results. Throughout we will use the term "gain" to denote the improvement (drop) in mean response time experienced by beneficiary jobs under cycle stealing, as compared with dedicated servers, and the term "pain" to refer to the increase in mean response time experienced by



(a) $E[X_B] = 1, E[X_D] = 1$ (b) $E[X_B] = 1, E[X_D] = 10$ (c) $E[X_B] = 10, E[X_D] = 1$

Fig. 7. *Validation of analysis against simulation. We fix $\rho_B = 0.9$, and vary ρ_L. (Donor response time is not shown for the case of $\rho_D = 1$, since the donor is unstable there.) $N_B^{th} = 3$ and $N_D^{th} = 2$. X_B and X_D are exponentially-distributed.*

donor jobs under cycle stealing as compared with dedicated servers:

$$\text{gain} = \frac{E[T_B]^{\texttt{Dedicated}}}{E[T_B]^{\texttt{CS}}} \quad \text{and} \quad \text{pain} = \frac{E[T_D]^{\texttt{CS}}}{E[T_D]^{\texttt{Dedicated}}},$$

where $E[T_B]^{\texttt{Dedicated}}$ refers to the mean response time of beneficiaries under dedicated servers and $E[T_B]^{\texttt{CS}}$ refers to the mean response time of beneficiaries under cycle stealing. $E[T_D]^{\texttt{Dedicated}}$ and $E[T_D]^{\texttt{CS}}$ are defined similarly. Observe that both pain and gain have been defined to range from 1 to ∞, where infinite gain corresponds to the situation where the mean response time under dedicated is infinite while it is finite under cycle stealing. In Sections 6.1-6.3 we fix the threshold values as $N_B^{th} = N_D^{th} = 1$ and study the effect of other parameters, and in Section 6.4 we study the effect of the threshold values. Due to limited space we typically only show a couple of options for each parameter. More graphs are available in the associated technical report [11].

6.1 Benefits of cycle stealing: wide range ρ_B

Cycle stealing is always a win when $\rho_B \geq 1$, but doesn't pay when $\rho_B \leq 0.5$. Figure 8 shows the mean response time for beneficiary jobs (top row) and donor jobs (bottom row) as a function of ρ_B, where ρ_D is low-to-medium ($\rho_D = 0.5$; columns 1 and 2) and medium-to-high ($\rho_D = 0.8$; columns 3 and 4). When $\rho_B \geq 1$ (and $\rho_D < 1$), cycle stealing can provide infinite gain to beneficiaries over dedicated servers, with comparatively little pain for the donors. This is because the *stability region* for the beneficiaries under cycle stealing is much greater than under dedicated servers. While factors such as

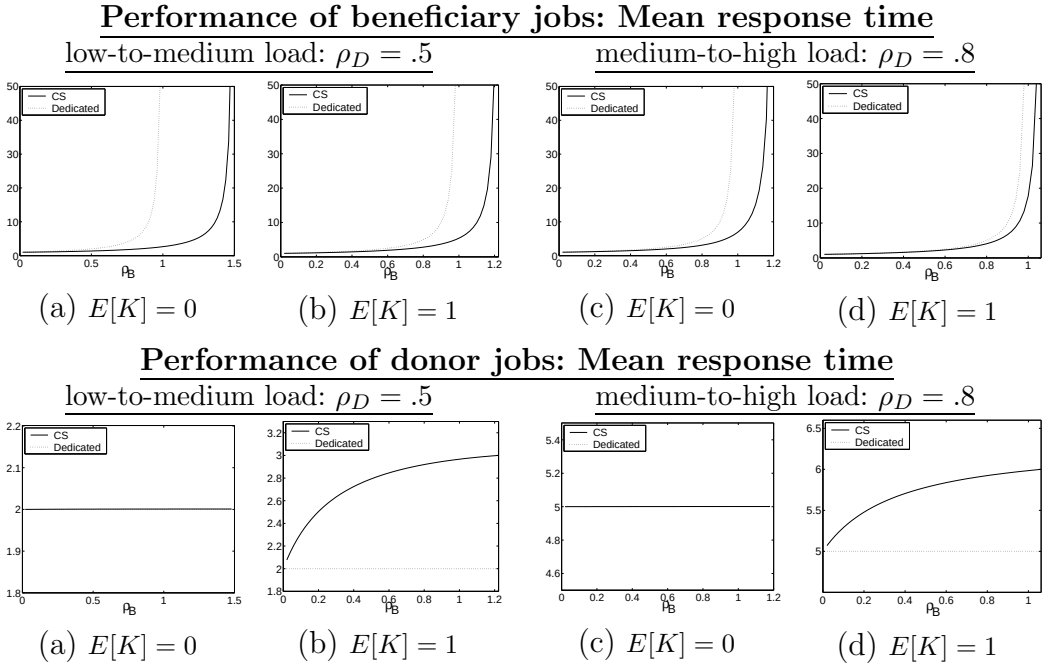


Fig. 8. *The mean response time for beneficiaries and donors as a function of ρ_B under cycle stealing and dedicated servers. In all figures X_B and X_D are exponential with mean 1; switching times are exponential with mean 0 or 1 as labeled.*

increased switching times and increased ρ_D do increase the mean response time of the beneficiary jobs, the gain is still infinite, and these factors are less important. We also see that the mean response time of donor jobs is bounded by the mean response time for an M/GI/1 queue with setup time K_{ba}. When $\rho_B \leq 0.5$, there is so little gain to the beneficiaries that cycle stealing with non-zero switching overhead doesn't pay. We therefore primarily focus the rest of the results section on the remaining case: $0.5 < \rho_B < 1$.

6.2 Benefit of cycle stealing: $.5 < \rho_B < 1.0$

For $.5 < \rho_B < 1$, cycle stealing has regions of high gain and low pain and also regions where the reverse is true. These regions depend on job sizes, switching times, and loads. In this section, we categorize performance into these gain/pain regions and also look at the *overall* mean response time (averaged over both beneficiary and donor jobs) to determine whether cycle stealing is "good" or "bad" overall. In general under higher ρ_B and lower ρ_D, cycle stealing is "good" overall, because the gain of the beneficiaries is so high in this region. We will find that when the switching times are short, cycle stealing leads to high gain and low pain. However long switching times can reverse this effect. More important than the absolute switching times are the switching times relative to the mean *donor* job size. We will find that the mean response time of the donors is sensitive to the

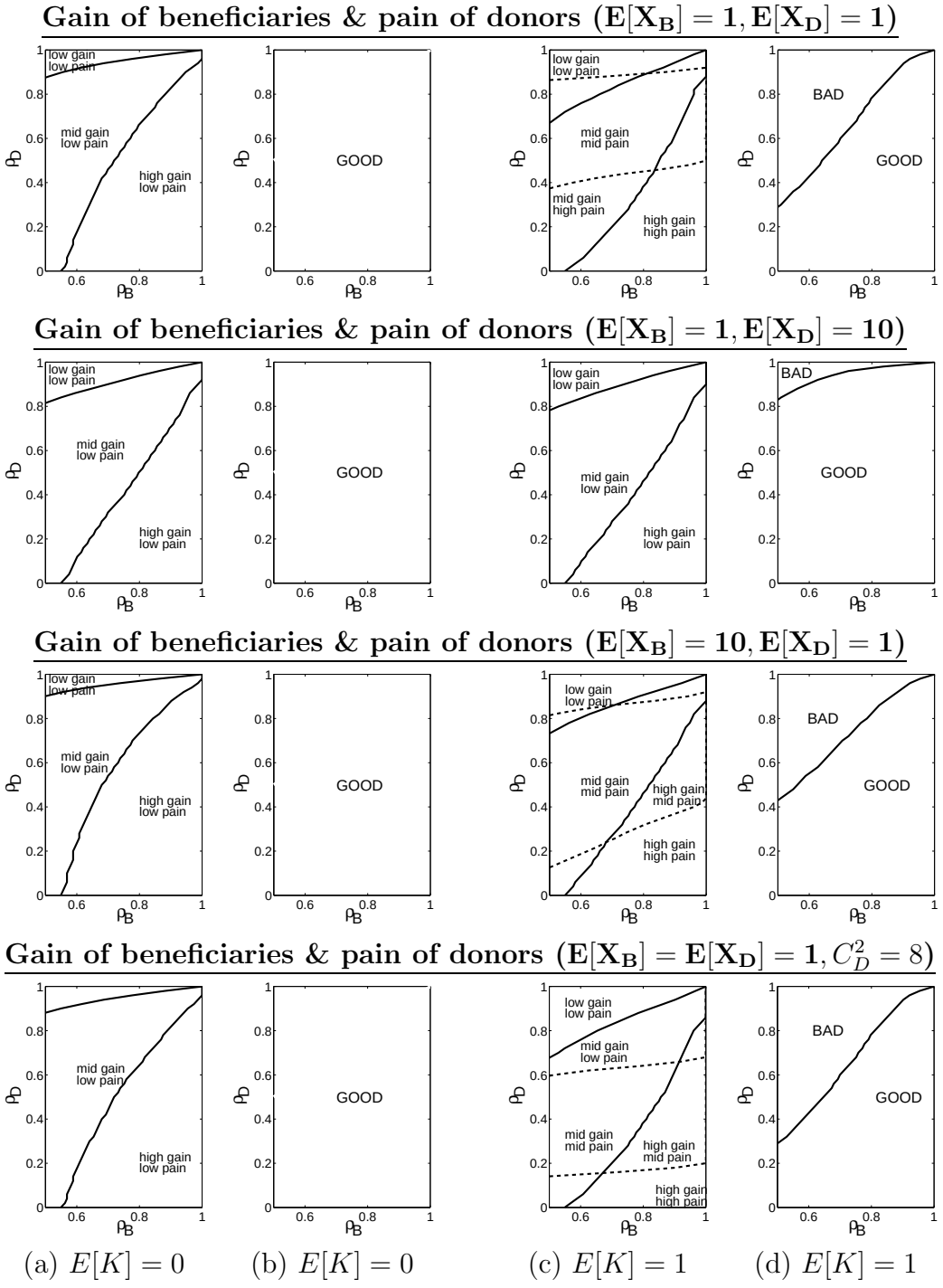
switching times, while surprisingly the mean response time of the beneficiaries is far less sensitive.

Figure 9 shows the gain of beneficiary jobs and the pain of donor jobs, where $.5 < \rho_B < 1$. The odd-numbered columns of Figure 9 divide the (ρ_B,ρ_D) space into regions of beneficiary gain and donor pain (low, mid, and high). We define *low gain* as a gain of 1.1 or less; *mid gain* as a gain of between 1.1 and 1.5; and *high gain* as a gain of over 1.5. Pain regions are defined similarly. While odd-numbered columns of Figure 9 consider the beneficiary and donor mean response time individually, the even-numbered columns of Figure 9 look at the overall mean response time and ask whether cycle stealing is "good" or "bad" with respect to the overall mean response time. The effect of mean job sizes is considered in the first three rows of Figure 9, where job sizes are exponential with mean 1 or 10. The effect of job size variability is considered in the fourth row of Figure 9, where the variability of donor job sizes is increased. The discussion of this fourth row is postponed to Section 6.3.

Consider first row 1 in Figure 9. Under zero switching time (a)-(b), all regions are low pain regions (in fact zero pain), and higher ρ_B yields higher gain for the beneficiaries. Non-zero switching times (c)-(d) create only slightly reduced gain for the beneficiaries, but they create pain for the donor jobs. When ρ_D is very low, the pain appears high, but this is primarily due to the fact that "pain" is relative to the mean response time under dedicated servers, which is clearly low for small ρ_D. Although not shown, we have also investigated longer switching times, and these lead to the same trend of slightly less gain for beneficiaries and significantly more pain for donors.

We now consider the effect of changes in job sizes. Row 2 of Figure 9 differs from row 1 only in X_D, which now has mean 10. The effect of this change is dramatic: now a switching time of 1 has almost no effect on either beneficiaries or donors. This makes sense since the setup time experienced by donor jobs is now relatively small compared to their size. Row 3 in Figure 9 differs from row 1 only in X_B, which now has mean 10. Comparing these rows, we see the increase in $E[X_B]$ has a surprisingly small effect on both beneficiaries and donors, as compared with increasing $E[X_D]$. This is because the donor still experiences the setup time, which has the same mean size as the donor job. We can conclude that cycle stealing is most effective when the switching time is small relative to the size of the *donor* jobs.

Focusing on columns 2 and 4 of Figure 9, which depict the effect on overall mean response time, we see that, for all rows, when the switching time is zero, cycle stealing always overwhelms the dedicated policy. When switching time is non-zero, cycle stealing is a good idea provided either ρ_B is high, or the switching time is short compared to X_D. These trends continue for longer switching times.



(a) $E[K]=0$ (b) $E[K]=0$ (c) $E[K]=1$ (d) $E[K]=1$

Fig. 9. *The gain of beneficiary jobs and pain of donor jobs (columns 1 and 3), and the effect of cycle stealing on the overall mean response time relative to dedicated servers (columns 2 and 4). In columns 1 and 3, solid lines delineate high/mid/low gain regions, and dashed lines delineate high/mid/low pain regions. X_B has an exponential distribution; X_D has an exponential distribution in rows 1-3 and a PH distribution with $C_D^2=8$ in row 4. The means of X_B and X_D are as labeled.*

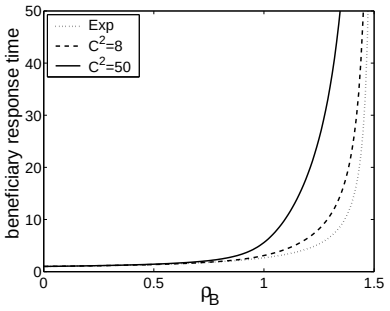


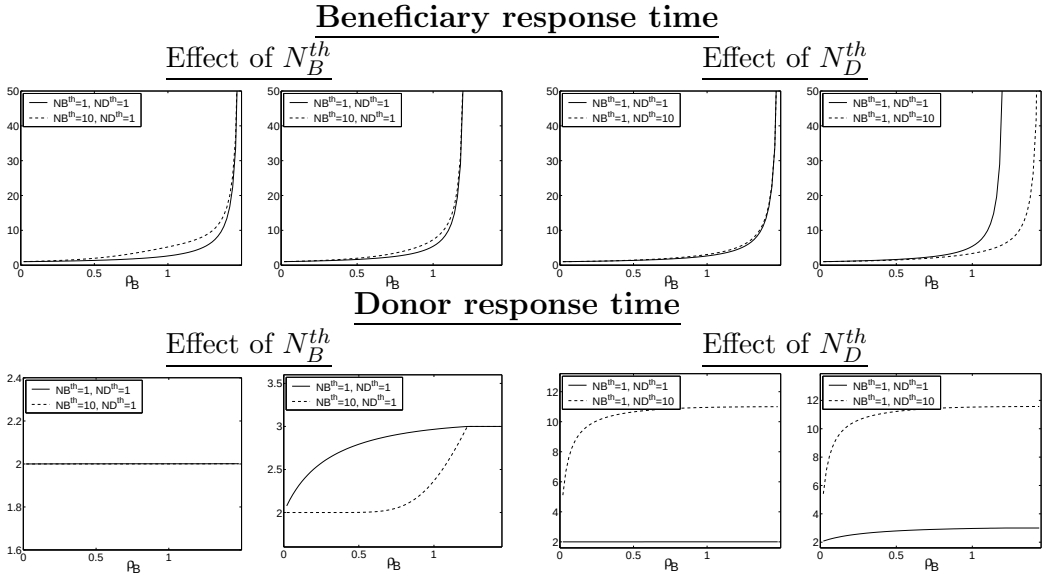
Fig. 10. *The mean response time for beneficiary jobs under different donor job size variability. We fix ρ_D at 0.5, and ρ_B varies from 0 to the stability condition of 1.5. The mean donor job size and beneficiary job size is 1, and switching time is zero.*

6.3 Effect of donor job size variability

For $.5 < \rho_B < 1$, we find variability of donor job sizes has very little effect on beneficiary mean response time. This finding surprised us; we expected the beneficiary to gain far less from the bursty help of a donor with irregular (highly variable) job sizes.

It seems intuitive that when donor job sizes are made more variable, two things should happen. (i) The donor pain should drop. This is because the donor mean response times will be higher overall, and so the relative pain will appear diminished. (ii) The beneficiary gain should drop. This is because high variability in the donor job sizes implies high variability in the length of the donor busy periods, which implies that the donor's visits to the beneficiary queue will be more irregular. Sporadic help should be inferior to regular help for the beneficiary. Figure 9 row 4 shows that hypothesis (i) is in fact true, while hypothesis (ii) is surprisingly false, at least for $\rho_B < 1$. Comparing row 1 (X_D has low variability: $C_D^2 = 1$) with row 4 (X_D has high variability: $C_D^2 = 8$), we see that there is no discernible difference in beneficiary performance.

To study this effect more closely, we next increase the variability in donor job sizes further. Figure 10 shows the mean response time of the beneficiary jobs under the case of zero switching time, when C_D^2 is 1, 8, or 50, and ρ_D is fixed at 0.5. We vary ρ_B from 0 to $\rho_B^{\max}$. As observed in Figure 9 row 4, the effect of the variability of X_D on the mean response time of X_B is small when $\rho_B < 1$, and negligible when $\rho_B < 0.75$. When $\rho_B > 1$ the effect of variability in donor sizes may be significant. A critical factor seems to be whether the beneficiary queue is stable in isolation; when this is not the case, high variability in donor visits leads to prolonged intervals of instability, which inflates the mean response time. This is the same phenomenon seen in [1] and [2].



(a) $E[K] = 0$ (b) $E[K] = 1$ (c) $E[K] = 0$ (d) $E[K] = 1$

Fig. 11. *The mean response time for beneficiaries and donors as a function of* ρ_B. *Graphs show the case of (i)* $N_B^{th} = N_D^{th} = 1$, *(ii)* $N_B^{th} = 10$ *and* $N_D^{th} = 1$, *and (iii)* $N_B^{th} = 1$ *and* $N_D^{th} = 10$. *In all figures* X_B *and* X_D *are exponential with mean 1. Switching times are exponential with mean 0 or 1 as labeled.* $\rho_D = 0.5$.

6.4 Effect of thresholds

The thresholds $\mathbf{N_B^{th}}$ and $\mathbf{N_D^{th}}$ have very different effects. In this section, we study the effect of threshold settings on performance. We will see that increasing N_B^{th} helps alleviate donor pain given nonzero switching time, without appreciably diminishing beneficiary gain. Thus, the optimal value of N_B^{th} tends to be well above 1. By contrast, increasing N_D^{th} increases beneficiary gain substantially (by increasing their stability region), but also increases donor pain. Overall, the impact of changes in N_D^{th} are much more dramatic than the impact of changes in N_B^{th}.

Figure 11 shows the mean response time for beneficiary jobs (top row) and the mean response time for donor jobs (bottom row) as a function of ρ_B for different threshold values. In the left half of the figure we study the effect of changing N_B^{th} from 1 to 10 as we hold N_D^{th} fixed at 1. In the right half of the figure we study the effect of changing N_D^{th} from 1 to 10 as we hold N_B^{th} fixed at 1. Throughout, X_B and X_D are exponential with mean 1 and we fix $\rho_D = 0.5$.

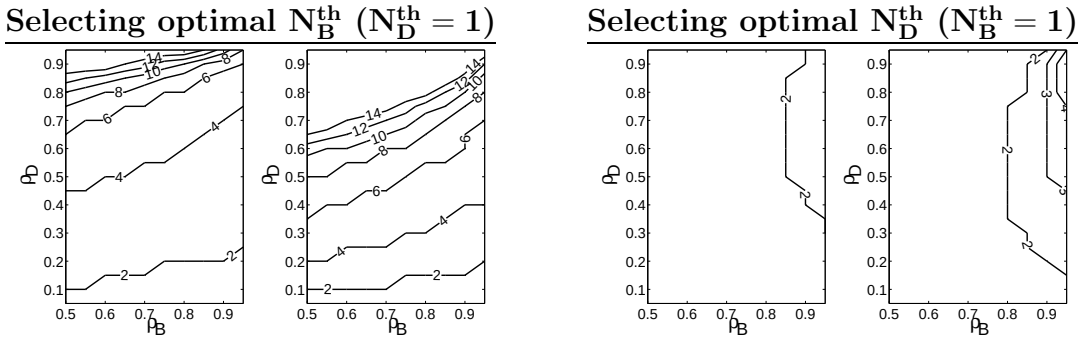
As N_B^{th} is increased from 1 to 10, Figure 11 shows only slightly higher response times for the beneficiary jobs. Recall that increasing N_B^{th} does not change the beneficiary stability region, although the beneficiary queue is helped less frequently. In fact, under longer switching times, the effect of raising N_B^{th} on beneficiary mean response time is even more negligible, since the decreased

frequency of helping beneficiaries is counteracted by the positive benefit of wasting less time on switching. We also see that increasing N_B^{th} creates less penalty for the donor, as the donor doesn't have to visit the beneficiary queue as frequently. Observe that when the switching times are nonzero, the donor mean response time is always bounded above by the mean response time for a corresponding M/GI/1 queue with setup time K_{ba}, and this bound is tight for all N_B^{th} values as ρ_B reaches its maximum, since the beneficiary queue always exceeds N_B^{th} in this case. We conclude that N_B^{th} has somewhat small impact; however higher values of N_B^{th} are more desirable for the system as a whole under longer switching time.

By contrast increasing N_D^{th} from 1 to 10 has dramatic effects. In general (assuming non-zero switching time) increasing N_D^{th} can drastically improve beneficiary response time. This result is not obvious, since increasing N_D^{th} allows the donor to spend more time at the beneficiary queue before leaving, but also means that when the donor leaves the beneficiary queue, the donor will be absent for a longer time (since more time is needed to empty the donor queue). Another positive effect of increasing N_D^{th} is less switching overall. In the end, it is the enlargement of the stability region due to higher N_D^{th} which substantially improves the beneficiary response time when the switching times are large and beneficiary load is high. When switching times are very short, increasing N_D^{th} only slightly worsens the mean response time for beneficiary jobs, as beneficiaries experience longer intervals between help. In all cases evaluated, increasing N_D^{th} results in much higher mean response times for donor jobs, since, for $N_D^{th} > 1$, the donor job arriving at an empty queue must wait for another $N_D^{th} - 1$ jobs to arrive before being served. We conclude that increasing N_D^{th} can have large impact, positive for the beneficiaries, but negative for the donors. Thus setting N_D^{th} is much trickier than N_B^{th}.

Finally we seek to determine good values for the thresholds, N_B^{th} and N_D^{th}, as a function of the system parameters. Above, we have already observed some characteristics of N_B^{th}. (i) Increasing N_B^{th} leads to lower gain for the beneficiaries and lower pain for the donors. (ii) Perhaps less obvious, the relative drop in gain for the beneficiaries is slight compared to the drop in pain for the donors. This points towards choosing a higher value of N_B^{th}. Thus, if the switching time is zero, the optimal N_B^{th} is 1 (or 0), since there is never any pain for the donors. Figure 12 (a) and (b) show optimal values of N_B^{th} for minimizing overall mean response time (over all jobs) as a function of ρ_B and ρ_D under various switching times when $N_D^{th} = 1$. The numbers on the contour curves show the optimal N_B^{th} at each load. For clarity we only show lines up to $N_B^{th} = 14$. The following additional characteristics of N_B^{th} are implied by the figure: (iii) the optimal N_B^{th} is an increasing function of ρ_D and a decreasing function of ρ_B; (iv) increasing the switching time increases the optimal N_B^{th}.

Figure 12 (c) and (d) show optimal values of N_D^{th} for minimizing overall mean



(a) $E[K] = 1$ (b) $E[K] = 2$ (c) $E[K] = 1$ (d) $E[K] = 2$

Fig. 12. *Graphs showing the optimal value of N_B^{th} and N_D^{th} with respect to overall mean response time, where X_B and X_D are exponentially distributed with mean 1.*

response time when $N_B^{th} = 1$. First observe that (i) under low ρ_D or low ρ_B the optimal N_D^{th} is 1. When ρ_D is low and $N_D^{th} > 1$, the pain for donor jobs is so huge that the optimal N_D^{th} is always 1. When ρ_B is low, the beneficiary gains little from increasing N_D^{th}, while the donor can have nonnegligible pain, which increases with N_D^{th}; hence the optimal N_D^{th} is always 1. The following characteristics of N_D^{th} are also implied by the figure: (ii) the optimal N_D^{th} is not a monotonic function of ρ_D, but is an increasing function of ρ_B; (iii) increasing the switching time increases the optimal N_D^{th}. Note that although the range of the optimal values of N_D^{th} is smaller than N_B^{th} in Figure 12, Figure 11 tells us that the performance effect of changing N_D^{th} on the mean response time of both beneficiary jobs and donor jobs is more significant than changing N_B^{th}.

7 Extensions and Current Work

This paper analyzes the mean response time under cycle stealing with switching times and thresholds, presenting many insights into the characteristics and performance of cycle stealing. Our analytical approach can be applied to other variants of cycle stealing models [11]. For example, we don't need to require that the donor switches back immediately when N_D^{th} is reached; we can allow the donor to first complete the beneficiary job in progress. Completing the beneficiary job obviates the need for checkpointing the job; however it sometimes reduces system performance, particularly when beneficiary jobs have higher variability than donor jobs. We can also handle the model of one beneficiary and two or more donor queues, where if i donors are helping, the beneficiary sees an M/GI/i queue. This extension also allows the different donors to have different switching thresholds and switching times. The case of multiple beneficiary queues does not seem readily solvable for the model in this paper, but has been solved for a related model [7]. Another interesting question involves servers that function as both beneficiaries and donors, as in [13]. Unlike our own model, in [13] the state of each processor is assumed to be

stochastically independent and identical to the state of the other processors. This means there is no need for a 2D-infinite chain, and no need for dimensionality reduction. In our, more complex model, the analysis of servers which function as both beneficiaries and donors is still open at the present time.

References

[1] S. Borst, O. Boxma, and P. Jelenkovic. Reduced-load equivalence and induced burstiness in GPS queues with long-tailed traffic flows. *Queueing Systems*, 43:274–285, 2003.

[2] S. Borst, O. Boxma, and M. van Uitert. The asymptotic workload behavior of two coupled queues. *Queueing Systems*, 43:81–102, 2003.

[3] J. Cohen and O. Boxma. *Boundary Value Problems in Queueing System Analysis*. North-Holland Publ. Cy., 1983.

[4] G. Fayolle and R. Iasnogorodski. Two coupled processors: the reduction to a Riemann-Hilbert problem. *Zeitschrift fur Wahrscheinlichkeitstheorie und vervandte Gebiete*, 47:325–351, 1979.

[5] R. D. Foley and D. McDonald. Exact asymptotics of a queueing network with a cross-trained server. In *Proceedings of INFORMS Annual Meeting*, pages MD06–2, October 2003.

[6] S. Fuhrmann and R. Cooper. Stochastic decompositions in the M/G/1 queue with generalized vacations. *Operations Research*, 33(5):1117–1129, 1985.

[7] M. Harchol-Balter, C. Li, T. Osogami, A. Scheller-Wolf, and M. S. Squillante. Cycle stealing under immediate dispatch task assignment. In *Proceedings of the Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 274–285, June 2003.

[8] A. Konheim, I. Meilijson, and A. Melkman. Processor-sharing of two parallel lines. *Journal of Applied Probability*, 18:952–956, 1981.

[9] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Modeling*. ASA-SIAM, Philadelphia, 1999.

[10] T. Osogami and M. Harchol-Balter. A closed-form solution for mapping general distributions to minimal PH distributions. In *Proceedings of the Performance TOOLS*, pages 200–217, 2003.

[11] T. Osogami, M. Harchol-Balter, and A. Scheller-Wolf. Analysis of cycle stealing with switching cost. Technical Report CMU-CS-02-192, School of Computer Science, Carnegie Mellon University, October 2002.

[12] T. Osogami, M. Harchol-Balter, and A. Scheller-Wolf. Analysis of cycle stealing with switching cost. In *Proceedings of the ACM SIGMETRICS*, pages 184–195, June 2003.

[13] M. S. Squillante and R. D. Nelson. Analysis of task migration in shared-memory multiprocessors. In *Proceedings of the ACM SIGMETRICS*, pages 143–155, May 1991.